Gradient Descent Style Leveraging of Decision Trees and Stumps for Misclassification Cost Performance

Mike Cameron-Jones

School of Computing,
University of Tasmania,
Launceston,
Tasmania,
Australia
Michael.CameronJones@utas.edu.au

Abstract. This paper investigates the use, for the task of classifier learning in the presence of misclassification costs, of some gradient descent style leveraging approaches to classifier learning: Schapire and Singer's AdaBoost.MH and AdaBoost.MR [16], and Collins et al's multiclass logistic regression method [4], and some modifications that retain the gradient descent style approach. Decision trees and stumps are used as the underlying base classifiers, learned from modified versions of Quinlan's C4.5 [15]. Experiments are reported comparing the performance, in terms of average cost, of the modified methods to that of the originals, and to the previously suggested "Cost Boosting" methods of Ting and Zheng [21] and Ting [18], which also use decision trees based upon modified C4.5 code, but do not have an interpretation in the gradient descent framework. While some of the modifications improve upon the originals in terms of cost performance for both trees and stumps, the comparison with tree-based Cost Boosting suggests that out of the methods first experimented with here, it is one based on stumps that has the most promise.

1 Introduction

Much work within the field of machine learning focusses on methods for learning classifiers for attribute value data. The methods learn classifiers from examples of known class with attribute value descriptions, attempting to predict well the class of new examples from their attribute value descriptions. Although the most common goal is to learn classifiers with high accuracy, in which case all mistakes are considered equally bad, mistakes can have different degrees of significance, e.g. for the owner of a new car, paying a year's theft insurance for a year in which the car is not stolen will (usually!) be a less expensive mistake, than not paying for the insurance in a year in which the car is stolen and not recovered. Thus some recent work has considered the issue of there being different misclassification costs associated with the different ways of misclassifying, e.g. [6] and [13]. In

some circumstances other forms of cost may be worth taking into account, see e.g. [22], but that is not pursued here, and henceforth costs will be assumed to be misclassification costs.

Recently, in the cost context there has been interest in the approach of learning classifiers that use the predictions of many component classifiers, e.g. Breiman's bagging [1] has been used for such problems in work involving the author [3], as have modifications of Freund and Schapire's boosting [9] by Fan et al [8], previously by Ting and Zheng [21], and subsequently by Ting individually [17, 18], and Ting and Witten's form of Wolpert's stacking [19, 20, 23] in work also involving the author [2]. This paper continues the theme of combined classifiers and costs, taking a further look at the boosting-style approaches, considering methods which are applicable to problems with two or more classes, not just two classes as considered in e.g. [8, 17]. In common with much of the previous work, the paper assumes a misclassification cost matrix representation of costs: for each class, there is a positive cost for each other class, representing the cost of misclassifying an item of the first class as being of the second class. The cost of correct classification is zero. The matrix is assumed to be available at the time the classifier is learned, but many of the methods here could be used in circumstances with differing costs per item, and only some need the cost information at learning time.

The original Adaboost method [9] and many of the subsequent variations aimed at accuracy maximisation have been interpreted as a form of gradient descent minimisation of a potential function, as in e.g. [7]; however, the previously successful boosting-style methods applicable to problems with misclassification costs and possibly more than two classes [21, 18], do not appear to be able to be interpreted in such a manner. This paper investigates experimentally the misclassification cost performance of some boosting-style methods previously proposed outside the cost context, and suggests some variations on them for the cost context while retaining the notion of the gradient of a potential function. The previous work on boosting-style approaches in the presence of costs has combined the predictions of many decision trees learned using (modified versions of) C4.5 [15], and this paper follows this, basing the underlying learner on C4.5, either as in the previous work, growing (and pruning with C4.5's standard criterion) a full tree, or restricting the tree grown to be a "stump", a tree with only one decision node and the leaves immediately below it. While the use of stumps has been investigated in the accuracy context [16], it has not previously been considered for problems with costs.

As Duffy and Helmbold remark [7], not all potential functions that have been used in variations on Adaboost lead to the formal PAC boosting property, and they use the term "leveraging" for the broader category of methods that in some sense leverage an underlying classifier learning method, hence our use of the term. However, where other authors have described their own method as a "boosting" method, even where it lacks the formal property, we may also use the term boosting.

The paper continues with a description of the methods considered, then the experiments and results, including a comparison with the previous methods, and finishes with some conclusions and suggestions for further work.

2 The Methods

This section describes the leveraging methods and variations that will be compared experimentally for misclassification cost performance. First the general framework will be outlined, then the specific methods within it.

2.1 General Framework

The general framework to be outlined here is based on the multiple class methods of Schapire and Singer [16], but restricted to the case where each item can have only one class. It leverages a learning method that will take a set of m instances, each of which has an attribute value description, a class label y in the set of possible labels L, and a set of weights, one for each possible class label, and return a classifier, a hypothesis h, which when given the description of an instance i and a class label l returns a prediction $h(i,l)$, in some cases this may be a +1/-1 prediction, in others a real valued prediction. Each such hypothesis is given a weight α, in some cases $\alpha = 1$.

If a series of classifiers, $h_1, h_2, \ldots h_t$ have been formed, the sum of the weighted classifier predictions for item i, label l, is $s(i,l) = \sum_{j=1}^{j=t} \alpha_j h_j(i,l)$. For a label l, the "margin" is $[y = l]s(i,l)$, where $[\ldots]$ is used to stand for +1/-1 for the enclosed expression being true / false respectively. Thus a positive margin for a label corresponds to a correctly signed $s(i,l)$, thresholding about zero.

The "gradient descent" view is based upon consideration of a potential function expressing the current extent of training error as a function of the margins of the training examples. The leveraging process attempts to minimise the function in a series of steps, in each of which it attempts to learn an underlying classifier that approximates the direction of the (negative) gradient with respect to the margins, then takes an appropriately sized step in that direction by adding the classifier's predictions (perhaps appropriately weighted) to the current combined classifier. (Note that some might consider the gradient descent perspective more specifically to apply to the two class case.)

Given a potential function defined in terms of the $s(i,l)$ in such a way that the gradient of the potential function with respect to the margins can be found, the general leveraging process here consists of repeating for a number of rounds, the following steps for the jth round:

1. Set weight for each item label pair (i,l) to be the negative of the gradient of the potential with respect to the margin of (i,l)
2. Normalise the weights so that the sum of the weights is the number of items
3. Learn h_j using modified C4.5
4. Calculate α_j

When the learning process is completed the resulting combined classifier can be used to predict the class of a new item from the s's for the new item's different labels.

The differences between the methods that are now to be described further is in the potential functions, the modification to the C4.5 learning and predicting process, the calculation of α_j, and the way that the s's are used to predict the label of a new item. These are now described for each of the methods that we consider.

2.2 Schapire and Singer's Multi-class AdaBoosts and Variations

The AdaBoost.MH and AdaBoost.MR methods of Schapire and Singer [16] have not previously been tested in the cost context, and they and some modifications aimed at the cost context are the main methods examined here. Both methods follow the previous boosting approaches in using exponential style potential functions, leading to exponential gradients and instance-label pair weights that can be updated simply (e.g. multiplicatively) in the implementation rather than calculating the gradient from scratch as the general framework might suggest. The original paper gives a fuller description than that here, though it is not approached from the gradient descent perspective.

The original work evaluated the methods using decision stumps, of a slightly different form to the C4.5 stumps we consider here, e.g. when testing a multiple-valued discrete attribute, the stumps here form one leaf per value, and use the C4.5 approach to handling missing values, (splitting items amongst branches), whereas the original work formed a stump with three leaves, one for a value chosen to test against, another for all other known values, and the last for missing values. This work also considers full trees, not just stumps, as used by Schapire and Singer in the original work.

AdaBoost.MH AdaBoost.MH, (Multi-class Hamming) is based upon the potential function $\sum_{i=1}^{i=m} \sum_{l \in L} \exp(-[y = l]s(i,l))$, reflecting a notion of Hamming style loss across instances and labels. Schapire and Singer suggest that for trees, each leaf can to some extent be considered as a separate classifier (making zero predictions on items that do not reach it) for the purposes of determining the appropriate prediction to make (i.e. step size to take) to minimise the potential function. The appropriate real valued leaf prediction for label l is $\frac{1}{2}\ln(\frac{W_+^l}{W_-^l})$ where W_+^l is the weight of items with class label l at the leaf and W_-^l is the weight of items with other class labels at the leaf. (In practice to avoid potentially infinite predictions a small constant weight is added to both Ws). Leaf predictions of this form lead to an appropriate splitting criterion based upon the Ws, which has been incorporated into our modified C4.5 along with the leaf predictions. The leaf predictions in effect render the αs redundant and they are set to 1. Predictions are made by the combined classifier by predicting the label with the greatest $s(i,l)$, (with simplistic tie-break here as in subsequent methods).

AdaBoost.MR In the single label case AdaBoost.MR (Multi-class Ranking) simplifies to AdaBoost.M2 [9], but this paper sticks with the term MR. The method is based upon the potential function $\sum_{i=1}^{i=m} \sum_{l \in L: l \neq y} \exp(s(i,l) - s(i,y))$, reflecting a notion of ranking loss with respect to the incorrect labels versus the correct label for each instance.

Schapire and Singer do not suggest an appropriate real valued leaf prediction, just the use of +1/-1 predictions of $[W_+^l > W_-^l]$ for label l. Leaf predictions of this form lead to an appropriate splitting criterion based upon locally maximising r, the weighted sum over instance label pairs of the correctness of the predictions (+1 for a correct prediction for the instance label pair, -1 for incorrect), and the splitting criterion and prediction method have been incorporated into our C4.5 code. Here Schapire and Singer address the issue of the appropriate step size at the level of the entire classifier, with α being $\frac{1}{2}\ln(\frac{1+r}{1-r})$. Predictions are made by the combined classifier by predicting the label with the greatest $s(i,l)$.

Non-uniform Initialisation The AdaBoost.M methods are designed to attach equal importance to each training example, and thus take no account of the relative importance in cost terms of getting different items right. Previous cost work, e.g. [12, 18] has used non-uniform weight initialisation to get the non-uniform misclassification costs taken into account by the classifier learning method. In this previous work, the classifier learning has used only one weight per item, and hence in problems with more than two classes, the different costs of making different forms of errors for an individual item have not been able to be expressed in the weighting – each item has been weighted proportionately to the sum of the costs of the way in which it can be misclassified. As this method in effect collapses a cost matrix to a cost vector, it will be referred to as the "vector" approach of initialisation.

A vector approach can be applied to the AdaBoost.M methods, by weighting each of the terms in their potential functions by the relevant costs. Letting c_i stand for the sum of the costs of the ways of misclassifying instance i, the potential functions become $\sum_{i=1}^{i=m} c_i \sum_{l \in L} \exp(-[y = l]s(i,l))$ (MH) and $\sum_{i=1}^{i=m} c_i \sum_{l \in L: l \neq y} \exp(s(i,l) - s(i,y))$ (MR). However, unlike the previous approaches, the Adaboost.M methods offer the possibility of the different costs of the different ways of misclassifying the one item being reflected in the potential function and hence learning process. Letting $c_{(i,l)}$ stand for the cost of misclassifying instance i as label l, the potential function for MR becomes $\sum_{i=1}^{i=m} \sum_{l \in L: l \neq y} c_{(i,l)} \exp(s(i,l) - s(i,y))$. While an appropriate cost weighting of the instance-label pairs with incorrect labels in MH seems straightforward, that for the correctly labelled pairs is less clear. Here we define $c_{(i,l)}$ to be c_i for the correct labels in the potential function $\sum_{i=1}^{i=m} \sum_{l \in L} c_{(i,l)} \exp(-[y = l]s(i,l))$. (Alternatives for the correct label case could be investigated).

Logistic-like Reinterpretation of the Outputs Although there do not appear to have been any experiments on the idea, Freund and Schapire [9] suggested

that the outputs of the two class Adaboost procedure could be used in a logistic form as probabilistic predictions, and Friedman et al [10] have also drawn the connection between logistic regression and Adaboost. Thus the possibility of using the $s(i,l)$ in a multi-class logistic style approach is considered here, predicting the probability of an item i being of label l as $\frac{exp(s(i,l))}{\sum_{j \in L} exp(s(i,j))}$. These probabilistic predictions can then be used with a misclassification cost matrix to estimate for each class the expected cost of predicting that class, and then the class with least estimated expected cost can be predicted.

Real Leaf Prediction for MR The real predictions from a single round for the MH method can potentially be different for each leaf and label, whereas the predictions for the MR method are just plus or minus the classifier weight α. The principle by which the weight α was chosen at the top level can be applied instead at the leaf level, yielding a (potentially) different (plus or minus) prediction for each leaf, though this does not go so far as the MH case where there is also the variation by label.

2.3 Logistic Regression Methods Based on Bregman Distances

Schapire and Singer have, along with Collins, placed their previous work in a mathematical framework based around minimisation of Bregman distances [4], and extended the work to include some related methods based explicitly upon potential functions using logistic regressions. The multi-class form is based on the (negative) log-prob potential function $-\sum_{i=1}^{i=m} \ln(\frac{exp(s(i,y))}{\sum_{j \in L} exp(s(i,j))})$. Like AdaBoost.MR, the function is based on the relative values of the s's and similarly to the situation with MR, +1/-1 predictions at leaves can be used, with the same form of splitting criterion and the same form of r-based weight α; however, the calculation of the instance-label pair weights at each round is no longer quite as simple. The use of the probability estimates is straightforward, and all the relevant code has been incorporated into our C4.5 based implementation.

The real leaf prediction mentioned for AdaBoost.MR can also be used with the logistic regression method.

An MH-like Variation Given the close connection between the differing methods, it seems natural to consider the possibility of a more MH-like form of the logistic regression method. A possibility examined here, based upon the method proposed by Collins et al for the two class case is to use the potential function $-\sum_{i=1}^{i=m} \sum_{j \in L} \ln(\frac{1}{1+exp(-[y=j]s(i,j))})$, which is MH-like in considering separately the potential for the problems of predicting each of the different labels. This has been implemented in the C4.5 based code, using the corresponding real valued predictions from MH, and an appropriate multi-class style logistic prediction.

3 Experiments

This section presents the results of the experiments on the methods previously described, examining in summary the relevant comparative performance of the different variations, and comparing in full some of the methods against the previously successful "Cost Boosting" methods [21, 18].

The commercial significance of true misclassification costs is such that, unfortunately for work in the area, very few data sets are made publically available with such costs. Hence in some of our previous work [2], and that of others, e.g. [21, 11], publically available data sets without given cost matrices have been used with a range of cost matrices generated for each data set, and the performance in terms of average misclassification cost being determined. For two class data sets the alternative approaches of examining ROC curves [14] or some other forms of cost curves such as those proposed by Drummond and Holte [5] would give a better representation of the results, but the methods do not scale simply to problems with more than two classes.

The 16 data sets chosen in our previous work [2] for their variety of characteristics and use in others' previous relevant work are used again here. A description, omitted in our previous work due to lack of space, is given in table 1. These are used as obtained from public sources except for the mushroom data set of which only a sample (10% rounded up) is used, as the full data set is uninterestingly straightforward for most methods.

Table 1. Data Sets

Name	Instances	Classes	Attributes Discrete / Continuous	Missing Values (%)
Abalone	4177	3	0 / 8	0.0
Colic	368	2	15 / 7	23.8
Credit-Australian	690	2	9 / 6	0.6
Credit-German	1000	2	13 / 7	0.0
Diabetes-Pima	768	2	0 / 8	0.0
Heart-Cleveland	303	5	7 / 6	0.2
Hypothyroid	3772	4	22 / 7	5.5
LED-24	200	10	24 / 0	0.0
Mushroom	813	2	22 / 0	1.4
Sick-Euthyroid	3772	2	22 / 7	5.5
Sonar	208	2	0 / 60	0.0
Soybean	683	19	35 / 0	9.8
Splice	3190	3	60 / 0	0.0
Tumor	339	22	17 / 0	3.9
Vowel	990	11	3 / 10	0.0
Waveform-40	300	3	0 / 40	0.0

As in the previous work, the random cost matrices are generated, like those of [21], to have zero diagonal elements as the cost of correct classifications is zero, one off-diagonal element is chosen at random to be 1, and the rest are then chosen uniformly from the integers 1 to 10.

Each average cost reported is an average over ten experiments. Each experiment consists of randomly generating a cost matrix and determining the performance of each learning method using a ten-fold cross validation. The same cost matrices and splits of the data for the cross validation were used for all learning methods, hence the results for any two learning methods on a data set can be thought of as paired results, in that only the learning methods differed.

When using leveraging methods, the issue of how many classifiers should be combined arises, as often performance will improve to a point then deteriorate with over-fitting. Most of the previous work with trees suggests that the significant performance benefits arise in about the first 10 rounds. Here we run the methods for 30 rounds with trees, choosing on which round to make a prediction on the basis of an internal hold-out procedure. One third of the training data is kept aside and the method run on the remaining two thirds for 30 rounds, evaluating the performance at each round, on the hold-out one third, then the method is run on all the training data for 30 rounds with the prediction being made at the round that appeared best on the basis of the internal hold out procedure. The round that is used for different forms of prediction may vary between them, e.g. although we do not report the results here, we measured the accuracy performance of the original methods, and the best round for accuracy performance determined by the internal hold out procedure may well be different from that for cost performance of the original method, which again may differ from that for cost performance with a multi-class logistic style probabilistic prediction, etc. For the stumps a similar approach was used but the methods were run for 60 rounds as the greater simplicity of stumps can cause more to be appropriate.

To reduce the number of results to be presented to more manageable proportions, the results of many of the comparisons between pairs of similar methods will be considered here simply in summary form as win-lose-tie performances, i.e. on how many data sets one was better, on how many the other was better, and on how many they tied, (to the accuracy to which we later report fuller figures).

The first issue that we examine is the use of the cost based weighting with the AdaBoost.M methods. We compare the vector style weighting with the raw unweighted methods, and the matrix style weighting with the vector style. The aggregate results over the MH and MR methods and over trees and stumps for each of these, i.e. 4 results for each dataset, show vector strongly improving upon raw by 56-4-4. The corresponding figures for the comparison between matrix and vector, ignoring the two class data sets, on which the methods are the same, are 28-6-2, fairly strongly supporting the view that the multiple weights per instance of the methods can be advantageous for the cost based weighting approach. (The artificial LED and waveform data sets cause most of the wins for the vector method, but it is not clear whether there is something interesting in this.)

The second issue that we examine is the use of the logistic style probabilistic predictions to make least expected cost predictions with the AdaBoost.M methods, instead of the simple original approach to predicting. Aggregating again over MR and MH and trees and stumps, shows the probabilistic predictions ahead by 58-3-3, a strong indication of improvement.

Given the two previous results, the question of which improvement upon the base approaches is the better, the matrix based weighting or the logistic style probabilistic predictions, arises. Aggregating over MR and MH and trees and stumps suggests some overall advantage to the logistic style predictions 36-23-5.

Using this logistic style prediction, there appears to be no genuine advantage to the real leaf prediction instead of the +1/-1 for the tree and stump MR approaches, ahead only 17-15-0.

A similar comparison of the explicit logistic regression method with AdaBoost.MR suggests a possible small advantage 21-10-1, though as this is composed of 14-1-1 for trees and 7-9-0 for stumps, there is a possibility that there is a dependency on the underlying classifier in there.

The use of the real leaf prediction with the logistic regression method is perhaps slightly ahead, 20-10-2, of the +1/-1, and the use of the more MH style potential function yields very similar overall performance to the original, 16-15-1 by comparison.

Thus overall the clear advantages by comparison against similar methods are to the matrix weighting method over the vector, and the use of logistic-style predictions for the AdaBoost.M methods instead of the simpler original method. The other approaches may be worth evaluating in individual applications, but do not appear to be major improvements overall.

The final question is how the methods compare against the previously proposed successful "Cost Boosting" methods of Ting and Zheng [21] and Ting [18]. Here we give the full results for some of the previously compared methods, focussing in terms of Adaboost on the MH method, which was originally suggested to be slightly better overall by Schapire and Singer [16]: MHMatT (AdaBoost.MH with matrix style weighting, using trees), MHMatS (as previous but stumps), MHPrT (AdaBoost.MH with logistic style probabilistic predictions, using trees), MHPrS (as before using stumps), LgPrT (Logistic regression with probabilistic predictions, using trees), LgPrS (as before using stumps), CBTZ (Ting and Zheng's method), CBT (Ting's method). Note that the "Cost Boosting" methods only use trees as they have not been designed to work with stumps. Table 2 shows the average misclassification costs per instance of these methods.

A comparison of the methods using the full trees shows that while the cost matrix style weighting and logistic style probabilistic predictions have been shown to improve upon the basic AdaBoost.MH method, they are inferior overall to the previous cost boosting methods. However, the use of Adaboost.MH with stumps and logistic style probabilistic predictions, while frequently producing very different results to the previous methods has comparable performance overall to each of the previous methods, being ahead of each in 12 domains, and marginally in front in terms of the geometric mean of the ratios of average costs.

Table 2. Full cost results for some methods

Data set	MHMatT	MHMatS	MHPrT	MHPrS	LgPrT	LgPrS	CBTZ	CBT
abalone	2.393	1.880	2.060	1.691	1.846	1.705	1.798	1.785
colic	0.555	0.531	0.598	0.531	0.573	0.570	0.616	0.626
credit-a	0.389	0.317	0.379	0.317	0.444	0.317	0.337	0.423
credit-g	0.708	0.606	0.729	0.610	0.740	0.598	0.608	0.618
diabetes	0.668	0.537	0.747	0.542	0.684	0.497	0.554	0.572
heart	2.501	2.088	2.281	1.947	2.295	1.845	2.171	2.198
hypothyroid	0.024	0.024	0.025	0.024	0.024	0.046	0.021	0.024
led	2.030	1.590	1.874	1.520	1.800	1.431	1.654	1.879
mushroom	0.013	0.018	0.015	0.020	0.041	0.022	0.021	0.016
sick	0.030	0.055	0.030	0.052	0.030	0.046	0.031	0.026
sonar	0.366	0.383	0.346	0.386	0.377	0.322	0.529	0.491
soybean	0.400	0.273	0.397	0.225	0.361	0.414	0.350	0.300
splice	0.163	0.148	0.161	0.138	0.161	0.162	0.198	0.228
tumor	3.201	2.929	2.681	2.468	2.647	2.438	2.698	2.987
vowel	0.427	0.966	0.423	0.975	0.186	2.111	0.766	0.484
waveform	1.168	1.029	1.160	0.972	1.041	0.799	1.068	1.072

Thus the appropriate gradient descent based method with stumps is competitive with the previous tree based approaches that do not fit the gradient descent framework. A check on accuracy performance suggests that this is not simply a matter of stumps being better than trees in all ways for the data sets used, as AdaBoost.MH's overall accuracy performance is better with trees than stumps, so the advantage of stumps lies in their suitability for the method of making probabilistic predictions, while not being superior in simple accuracy terms. (A check on the number of classifiers chosen by the internal validations suggests that the stump based approaches might benefit from more rounds in some cases, especially on the vowel data set where the average number of classifiers chosen is close to the maximum possible – some further experiments will be conducted on this.)

4 Conclusions and Further Work

This paper has examined the use in the misclassification cost context of some gradient descent leveraging methods using decision trees and stumps as the underlying base classifiers, experimentally comparing on 16 data sets the cost performance of the original methods and some modifications of them, and previous cost boosting approaches. The results show that the use of multiple weight per item methods enables the use of a more matrix style weighting method that performs better than previous weighting methods that collapsed the matrix to a vector. The results show that the use of a multi-class logistic probabilistic prediction from the leveraging methods performs better than the simple original prediction methods intended for the accuracy context. When compared against

previously proposed cost boosting methods using trees, the performance of one of the new stump based methods is competitive overall in cost terms. Accuracy performance results of the same method with trees suggest that the use of stumps may be particularly suited to the probabilistic prediction approach, as the trees perform better in accuracy terms.

Given the gap in performance between the tree based gradient descent approaches and the previous cost boosting approaches, an interesting possible direction to pursue seems to be in the creation of potential functions that better reflect the cost task in some way, and we are looking at some possibilities of this form at present. Given our previous results on stacking different forms of classifier [2], another fairly straightforward issue to investigate is whether the successful boosted stumps method constitutes a useful base classifier for the stacking approach.

Although this paper has put forward some successful cost based modifications to previous gradient descent leveraging methods, and experimentally demonstrated their potential, there are still some interesting possibilities to pursue in this area.

Acknowledgements

The work described in this paper was started while the author was on study leave at AT&T Labs Research, whom the author wishes to thank, particularly Rob Schapire and Michael Collins. Thanks are also due to the UCI repository maintainers, and the contributors, e.g. R. Detrano for the Cleveland data, and M. Zwitter and M. Soklic for the primary tumour data which was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Finally thanks are due to Robyn Gibson for comments on the paper.

References

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[2] M. Cameron-Jones and A. Charman-Williams. Stacking for misclassification cost performance. In *Advances in Artificial Intelligence: 14th Biennial Conference Canadian Society for Computational Studies of Intelligence, AI2001*, pages 215–224. Springer Verlag, 2001.

[3] M. Cameron-Jones and L. Richards. Repechage bootstrap aggregating for misclassification cost reduction. In *PRICAI'98: Topics in Artificial Intelligence – Fifth Pacific Rim International Conference on Artificial Intelligence*, pages 1–11. Springer Verlag, 1998.

[4] M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 158–169. Morgan Kaufmann, 2000.

[5] C. Drummond and R.C. Holte. Explicitly representing expected cost: An alternative to roc representation. Technical report, University of Ottawa, 2000.

[6] C. Drummond and R.C. Holte. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pages 239–246. Morgan Kaufmann, 2000.

[7] Nigel Duffy and David Helmbold. Potential boosters? In *Advances in Neural Information Processing Systems 12*, pages 258–264. MIT Press, 2000.

[8] W. Fan, S.J. Stolfo, J. Zhang, and P.K. Chan. Adacost: Misclassification cost-sensitive boosting. In *Machine Learning: Proceedings of the Sixteenth International Conference (ICML '99)*, pages 97–105, 1999.

[9] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression. Technical report, Stanford University, 1998.

[11] D. Margineantu. Building ensembles of classifiers for loss minimisation. In *Proceedings of the 31st Symposium of the Interface: Models, Prediction and Computing*, pages 190–194, 1999.

[12] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning (ML94)*, pages 217–225. Morgan Kaufmann, 1994.

[13] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[14] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML'98)*. Morgan Kaufmann, 1998.

[15] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. The Morgan Kaufmann Series in Machine Learning.

[16] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

[17] K.M. Ting. A comparative study of cost-sensitive boosting algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pages 983–990. Morgan Kaufmann, 2000.

[18] K.M. Ting. An empirical study of metacost using boosting algorithms. In *Proceedings of the Eleventh European Conference on Machine Learning (ECML-2000)*, pages 413–425. Springer Verlag, 2000.

[19] K.M. Ting and I.H. Witten. Stacked generalization: when does it work? In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 866–871. Morgan Kaufmann, 1997.

[20] K.M. Ting and I.H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

[21] K.M. Ting and Z. Zheng. Boosting trees for cost-sensitive classifications. In *Machine Learning: ECML-98: Proceedings of the Tenth European Conference on Machine Learning*, pages 190–195. Springer-Verlag, 1998.

[22] P.D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

[23] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.